



Universitatea Tehnică „Gheorghe Asachi” din Iași

Facultatea de Automatică și Calculatoare

Domeniul: Calculatoare și Tehnologia Informației



Online Shopping Application

“BD Express”

Proiect la disciplina
Baze de Date

Student: Enachi Vasile

Anul: 3

Grupa: 1308B

Coordonator: Cristian Buțincu

Capitolul 1. Introducere

Proiectul dat imită funcționalitatea unei baze de date în sfera shopping-ului online prin intermediul unui API (application programming interface).

API-ul din proiectul curent oferă utilizatorului posibilitatea de a naviga în tabela produselor, magazinelor și distribuitorilor. Pentru început user-ul este redirecționat pe pagina de ‘login’, unde acesta se poate autentifica cu profilul pe care îl deține sau poate crea un profil nou.

În aplicație există 4 tipuri de utilizatori, în funcție de tipul account-ului, fiecare utilizator dispune de roluri și privilegii diferite:

- client
- admin shop
- admin shipping
- global admin

1. Client

Utilizatorul cu tipul de account *client* beneficiază de posibilitatea de a căuta în tabelul de magazine, produse și distribuitori având la dispoziție mai multe fielduri de căutare în funcție de necesitate (nume, preț, preț range, id, locație ș.a.).

2. Admin Shop

Pe lângă avantajele pe care le are *clientul*, *administratorul de magazine* dispune de posibilitatea de a insera, modifica sau șterge înregistrări din tabelul **Shops** și tabelul **Products**.

3. Admin Shipping

Utilizatorul cu tipul account-ului *administrator de distribuitori* dispune de avantajele care le are *clientul*, precum și posibilitatea de inserare, modificare și stergere a înregistrărilor din tabelul **Shipping_Methods**.

4. Global Admin

Utilizatorul cu account-ul de tip *admin global* dispune de avantajele pe care le are atât *admin_shop*-ul cât și *admin_shipping*-ul. Pe lângă aceasta, dispune și de o pagină proprie de vizualizare a tuturor userilor, cu informațiile de rigoare (Nume, prenume, locație, email, phone, username, password) cu precizarea că acesta nu va putea vizualiza parolele celorlalți useri care au tipul de account *global_admin*.

Din scopuri pur didactice, asignarea tipurilor de account-uri se face la crearea account-ului userilor din interfața grafică.

Capitolul 2. Tehnologiile folosite pentru front-end și back-end

Baza de date folosită în această aplicație este SQLPlus. Pentru partea de front-end s-a folosit librăria tkinter din python.

Frame-ul principalul denumit container se află în clasa BdGui. Această clasă deține un dicționar cu alte 7 frame-uri, care în funcție de utilizator pot fi invocate prin intermediul funcției **tkraise()**. Relația dintre clasele LoginPage, SignUpPage, AdvancedAdminPage, HomePage, ShopPage, ShippingPage, ProductPage este de agregare față de clasa BdGui.

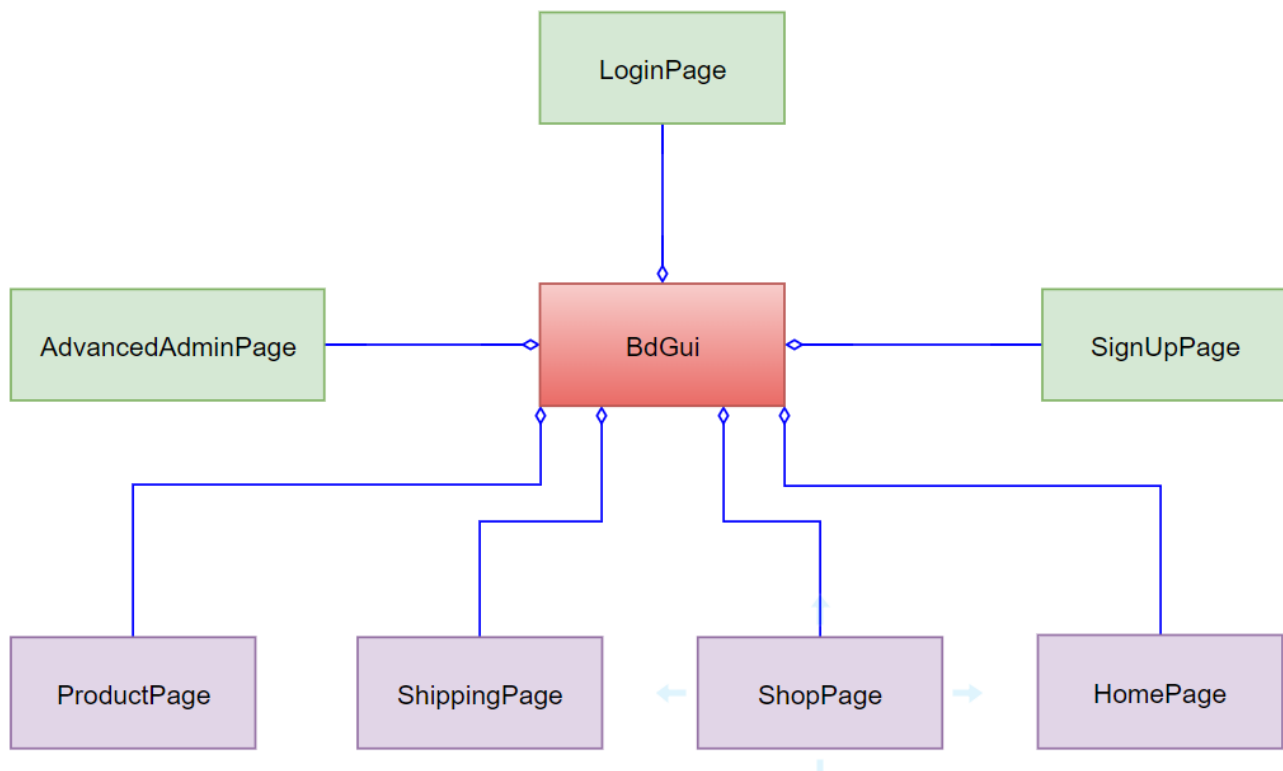


Fig 2.1. Diagrama de clase pentru paginile aplicației

Pentru vizualizarea înregistrărilor, a fost creată o clasă TableFrame care conține un atribut de tipul **tkinter.TreeView**. Clasele AdvancedAdminPage, HomePage, ShopPage, ShippingPage, ProductPage dețin ca atribut clasa TableFrame realizând o relație de compoziție.

Pentru introducerea datelor au fost folosite widget-urile tkinter.Entry, tkinter.Combobox și tkinter.CheckBox din cadrul librăriei tkinter. Restul diagramei de clase este prezentată în Fig. 2.

Criptarea și decriptarea parolei

Pentru criptarea și decriptarea parolei sunt folosite metodele encrypt și decrypt din clasa Cipher, care utilizează clasa Fernet din librăria cryptography. Criptarea și decriptarea parolei este necesară atunci când baza de date nu rulează la nivel local, în cazul de față operațiile date sunt făcute în scop academic la logare și creare de account, înainte de executarea instrucțiunilor sql. Criptarea și decriptarea se face folosind o cheie, în cazul de față cheia fiind: “**my deep darkest secret is secure**” codificată într-un șir de octeți folosind librăria base64 (clasa Fernet accept doar chei '32 url-safe base64-encoded bytes').

Parola este menținută în baza de date sub forma unui hash generat de funcția criptografică md5 folosind librăria hashlib.

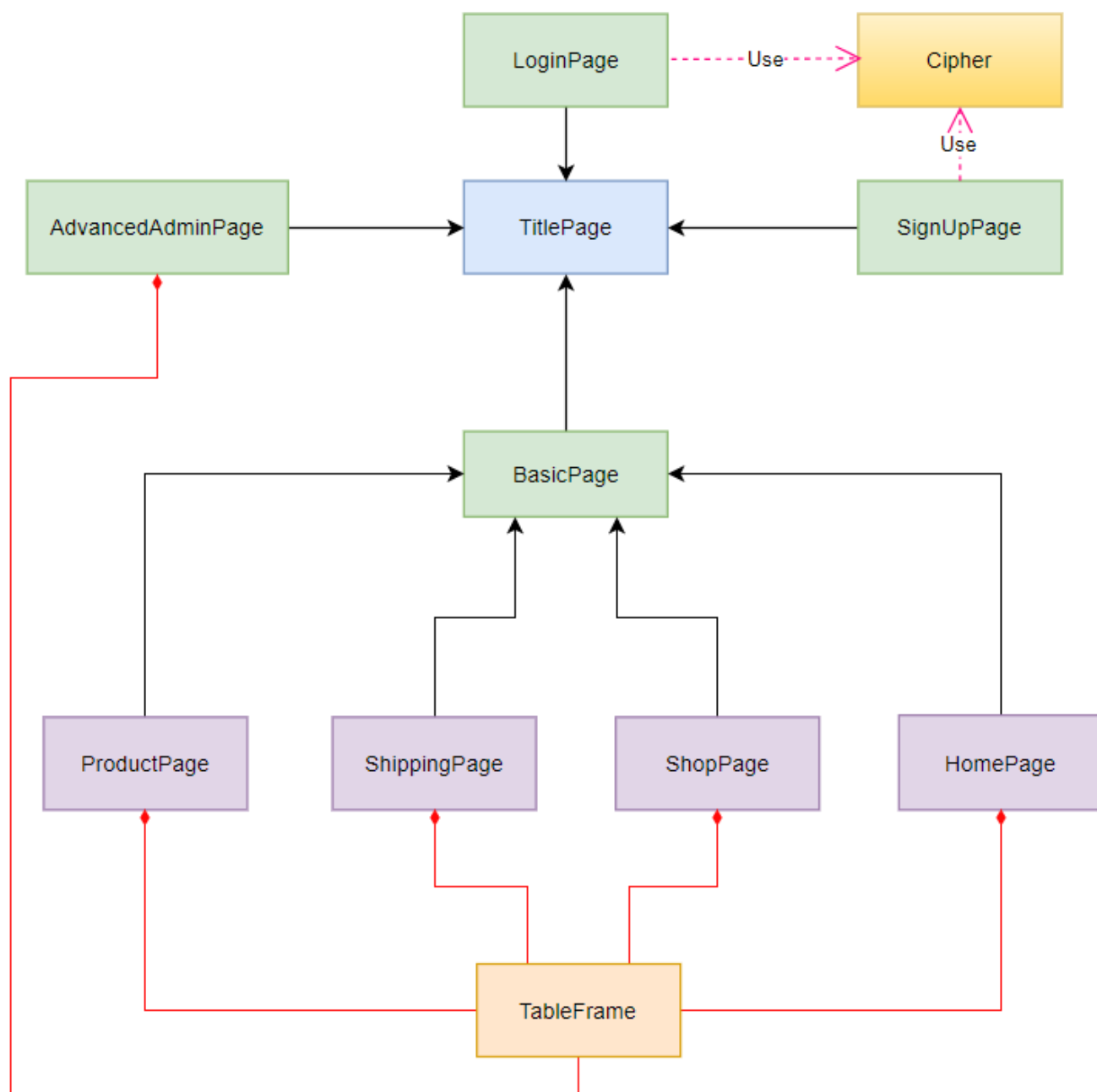


Fig 2.2. Diagrama de clase extinsă, fără clasa BdGui

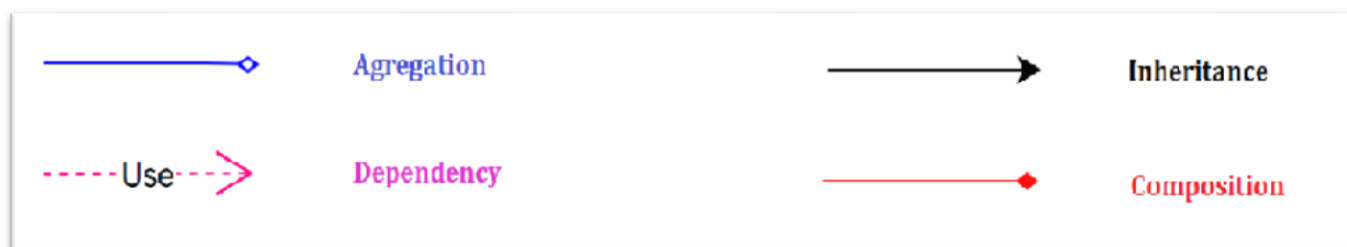


Fig 2.3. Legendă

Capitolul 3. Structura și inter-relaționarea tabelelor

Structura tabelor a fost creată așa încât să satisfacă unul din multiplele modele real a bazelor de date pentru shopping-ul real.

- Utilizatorilor le este permis să dețină o singură locație (la care va fi livrată comanda). Într-o locație pot exista mai mulți utilizatori (One-To-Many)
- Magazinelor le este permis să dețină o singură locație. Într-o locație pot exista mai multe magazine (One-To-Many)
- Într-un magazin pot exista mai multe produse, iar produsul înregistrat în baza de date aparține unui singur magazin (One-To-Many)
- O comandă poate conține doar un singur produs și un produs poate face parte din mai multe comenzi (One-To-Many). O comandă poate fi efectuată doar de un singur utilizator și un utilizator poate avea mai multe comenzi (One-To-Many). Acest lucru permite să se stabilească indirect o relație Many-To-Many între tabelul Products și App_Users (un utilizator poate comanda de mai multe ori același produs și același produs poate fi comandat de mai mulți utilizatori diferiți)
- O comandă poate fi livrată prin intermediul unui singur furnizor de livrări
- Un utilizator îi este asignat un singur account și un account poate fi deținut de un singur utilizator (relație One-To-One)

Ținând cont de precizările de mai sus, Diagrama ER (Entity-Relation) construită după modelul Crow's Foot Notation este reprezentată în figura Fig. 3.1.

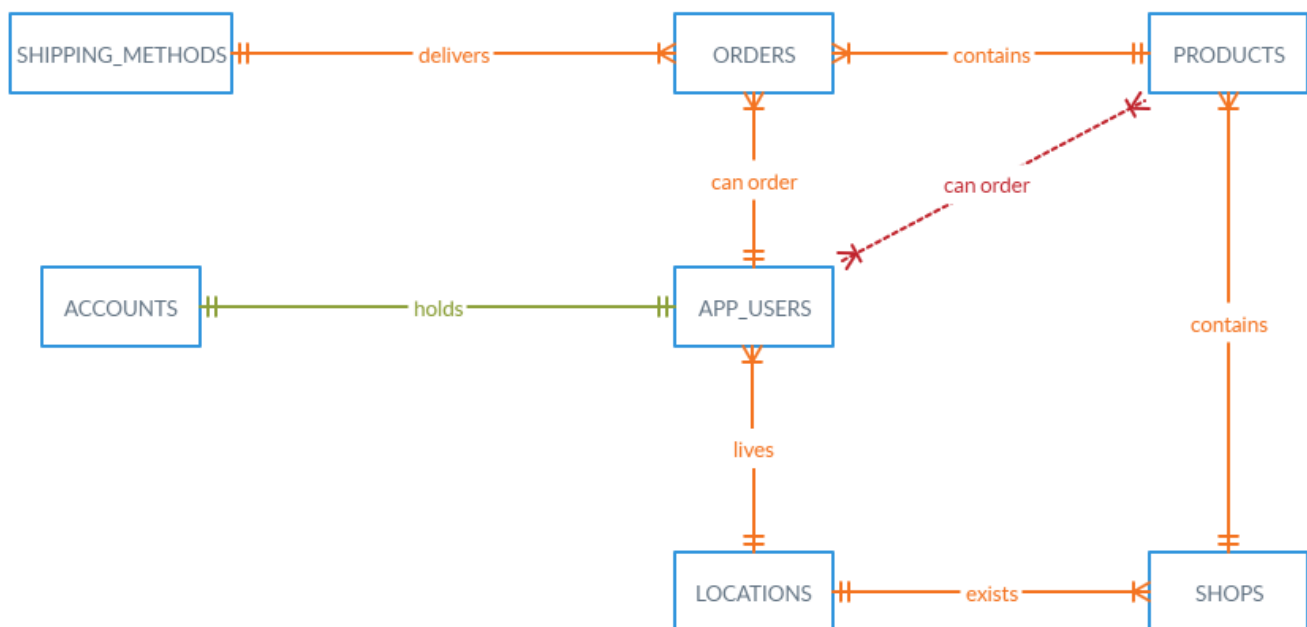


Fig. 3.1 Diagrama ER (Crow's Foot Notation)

Capitolul 4. Descrierea constrângerilor folosite

Constrângerile, atributele și cheile tabelelor sunt reprezentate în figura Fig.4.1.

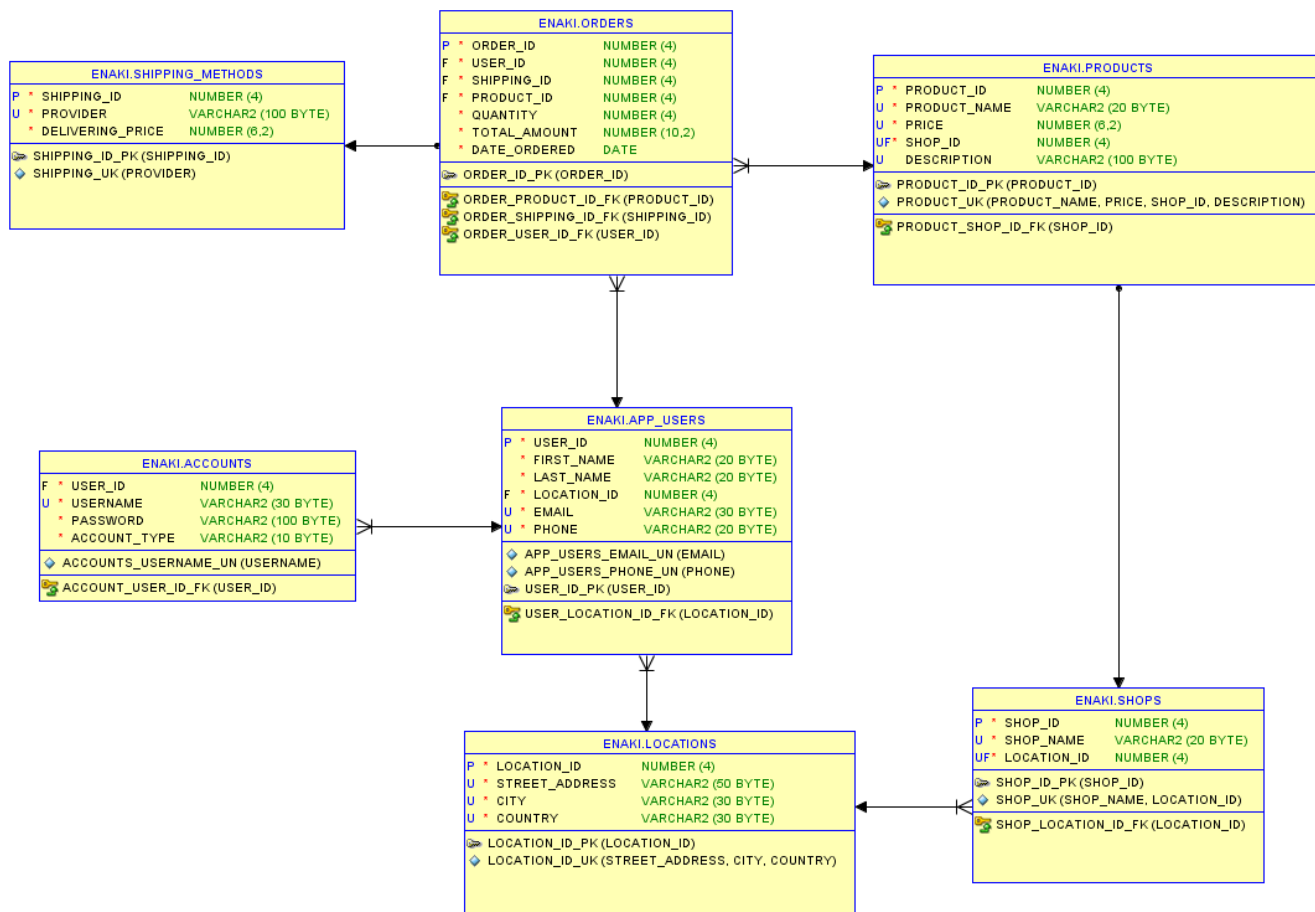


Fig. 4.1 Atributele, constrângerile și cheile tabelor

Pentru toate atributele din fiecare tabel, cu excepția atributului **Description** din tabelul **Products** s-a impus constrângerea de tip check – Not Null.

- **Shipping_Methods**

Primary Key: shipping_id

Constrângere Unique: provider (un furnizor are un nume unic)

Constrângere Check: delivering_price (prețul trebuie să fie pozitiv)

- **Orders**

Primary Key: order_id

Constrângere Check: quantity, total_amount (cantitatea și suma totală trebuie să fie pozitive)

Foreign Keys: user_id (app_users), shipping_id (shipping_methods), product_id (products)

- **Locations**

Primary Key: location_id

Constrângere Unique: (street_address, city, country)

- **Shops**

Primary Key: shop_id

Constrângere Unique: (shop_name, location_id)

Foreign Keys: location_id

- **Products**

Primary Key: product_id

Constrângere Unique: (product_name, price, shop_id, description)

Constrângere Check: price (prețul trebuie să fie pozitiv)

Foreign Keys: shop_id

- **Accounts**

Primary Key: user_id

Constrângere Check: password (parola trebuie să aibă minim 6 caractere)

Constrângere Unique: username (un account trebuie să aibă un nume de utilizator unic)

Foreign Keys: user_id

- **App_Users**

Primary Key: user_id

Constrângere Unique: email, phone

Foreign Keys: location_id

Normalizare

Toate tabelele sunt aduse cel puțin la a 3 formă normală.

- Un atribut conține valori atomice din domeniul său și nu grupuri de valori
- Atributele non-cheie depind de toate cheile candidat
- Atributele non-cheie nu sunt tranzitiv dependente de cheile candidat

Forma normală Boyce-Codd este de asemenea asigurată, întrucât:

- Pentru o dependență $X \rightarrow Y$, rezultă că X – super cheie

Capitolul 5. Descrierea modalității de conectare la baza de date

Conectarea la baza de date a fost făcută prin intermediul librăriei cx_Oracle.

```
def run_query(self, query):
    cursor = self.conn.cursor()
    cursor.execute(query)
    try:
        query_results = [row for row in cursor]
        self.conn.commit()
        cursor.close()
    except cx_Oracle.InterfaceError:
        self.conn.commit()
        cursor.close()
        return None
    return query_results
```

Metoda run_query execută o interogare SQL și returnează rezultatele într-o listă de tuple. Această metodă aparține clasei BdGui.

Informații auxiliare

În program sunt verificate datele introduse de utilizator conform constrângerilor impuse în baza de date. Nerespectarea datelor de intrare este atenționată utilizatorului prin intermediul unei ferestre de tip Pop-Up, folosind widgetul tkinter.messagebox.

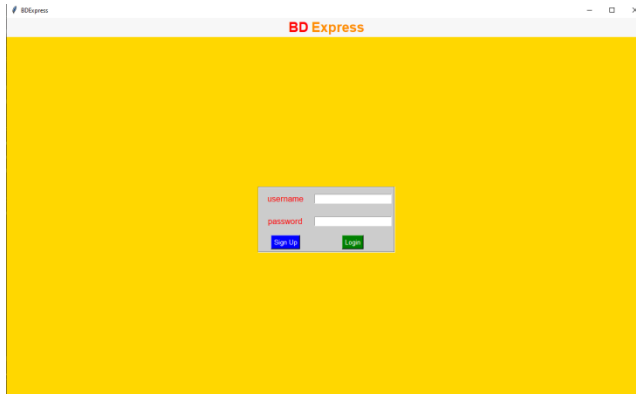
În clasa TableFrame a fost definită o metodă de sortare a datelor din interfață crescător sau descrescător, la apăsarea coloanei cu mouse-ul de către utilizator.

Primary Key-urile de tip id sunt generate de baza de date după tipul AUTO-Increment. Pentru acest lucru se crează o secvență de increment, care se folosește într-un trigger.

```
--
74 CREATE SEQUENCE location_sequence_incrementer START WITH 1 INCREMENT BY 1;
--
81 CREATE OR REPLACE TRIGGER location_inc_on_insert_trigger
82 BEFORE INSERT
83 ON locations
84 FOR EACH ROW
85 BEGIN
86     :NEW.location_id := location_sequence_incrementer.nextval;
87 END;
88 /
--
```


Capitolul 6. Capturi de ecran

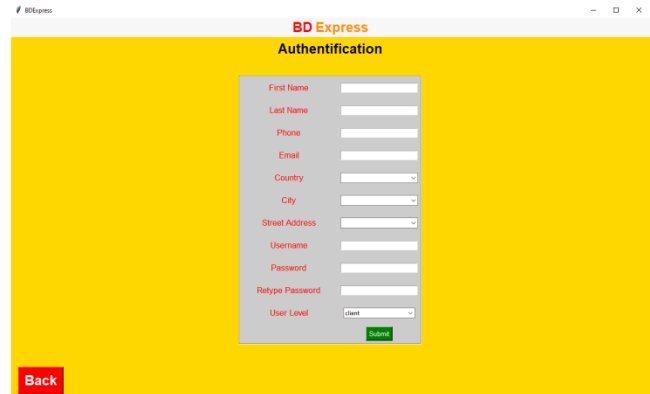
- Interfața grafică



BD Express

username

password



BD Express

Authentication

First Name

Last Name

Phone

Email

Country

City

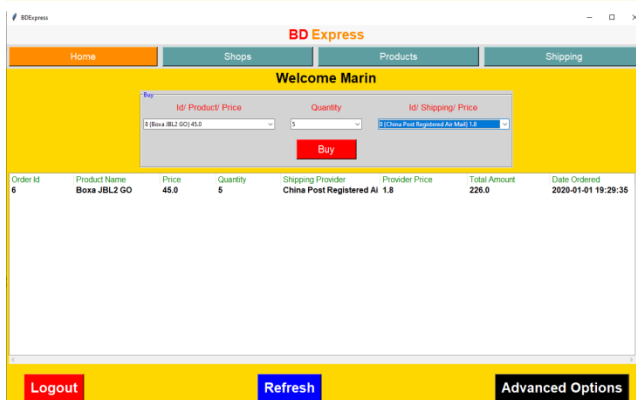
Street Address

Username

Password

Retype Password

User Level



BD Express

Home Shops Products Shipping

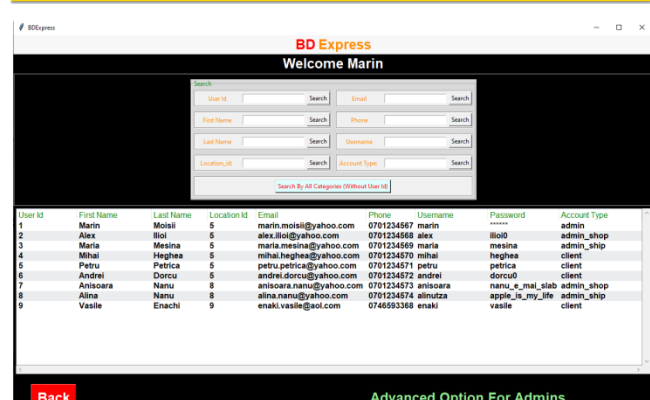
Welcome Marin

Buy

Id Product Price Quantity Id Shipping Price

1 (Show JBL2 GO) 45.0 1 2 (China Post Registered Air Mail) 1.8

Order Id	Product Name	Price	Quantity	Shipping Provider	Provider Price	Total Amount	Date Ordered
6	Boxa JBL2 GO	45.0	5	China Post Registered Ai	1.8	226.0	2020-01-01 19:29:35



BD Express

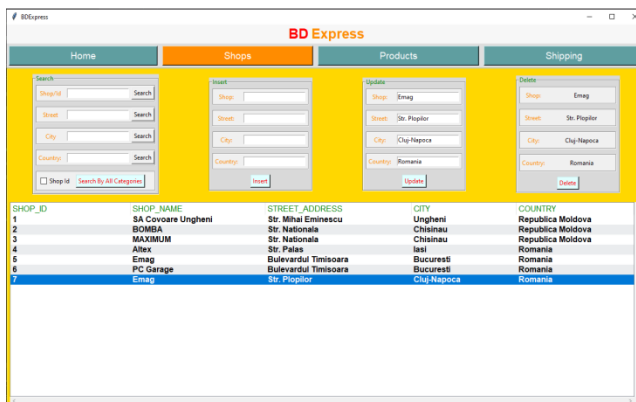
Welcome Marin

Search

User Id Search First Name Search Last Name Search Location Id Search Account Type Search

Search By All Categories (Without User Id)

User Id	First Name	Last Name	Location Id	Email	Phone	Username	Password	Account Type
1	Marin	Moisil	5	marin.moisil@yahoo.com	0701234567	marin	marin	admin
2	Alex	Ilioi	5	alex.ilioi@yahoo.com	0701234568	alex	alex	admin_shop
3	Maria	Mesina	5	maria.mesina@yahoo.com	0701234569	maria	marina	admin_shop
4	Mihai	Heghea	5	mihai.heghea@yahoo.com	0701234570	mihai	heghea	client
5	Petru	Petrica	5	petru.petrica@yahoo.com	0701234571	petru	petrica	client
6	Andrei	Dorciu	5	andrei.dorciu@yahoo.com	0701234572	andrei	dorciu	client
7	Anisoara	Nanu	8	anisoara.nanu@yahoo.com	0701234573	anisoara	anisoara	admin_shop
8	Alina	Nanu	9	alina.nanu@yahoo.com	0701234574	alina	alina	admin_shop
9	Vasile	Enachi	9	enaki.vasile@aol.com	074693368	enaki	vasile	client



BD Express

Home Shops Products Shipping

Search

Shop Id Search Street Search City Search Country Search

SHOP_ID	SHOP_NAME	STREET_ADDRESS	CITY	COUNTRY
1	SA Covoare Ungheni	Str. Mihai Eminescu	Ungheni	Republica Moldova
2	BOMBIA	Str. Nationala	Chisinau	Republica Moldova
3	MAXIMUM	Str. Pales	Chisinau	Republica Moldova
4	Alex	Bulevardul Timisoara	Bucuresti	Romania
5	Enag	Bulevardul Timisoara	Bucuresti	Romania
6	PC Garage	Bulevardul Timisoara	Bucuresti	Romania
7	Enag	Str. Poplar	Cluj Napoca	Romania



BD Express

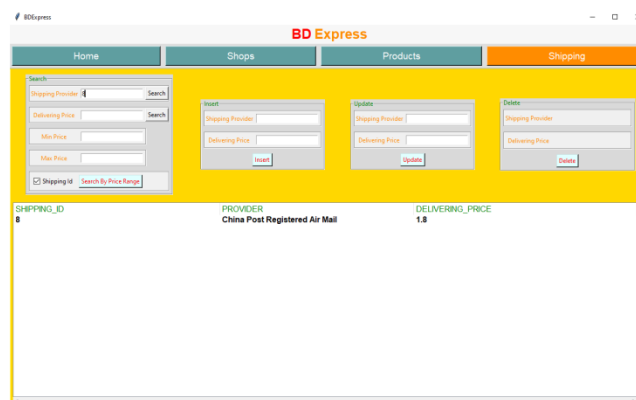
Home Shops Products Shipping

Search

Product Id Search Price Search

New Price Max Price

PRODUCT ID	PRODUCT NAME	PRICE	SHOP ID	DESCRIPTION
1	Covor Traditional	20.0	1	None
2	Covor Modern Vintage	40.0	1	None
3	Cuptor cu microunde	35.2	2	None
4	Cuptor cu microunde	48.0	3	None
5	Masina de spalat	55.0	3	None
6	Geanta Deli	20.0	6	None
7	Geanta Deli	35.0	7	None
8	Boxa JBL2 GO	45.0	7	None



BD Express

Home Shops Products Shipping

Search

Shipping Provider Search Delivering Price Search

Min Price Max Price

SHIPPING_ID	PROVIDER	DELIVERING_PRICE
6	China Post Registered Air Mail	1.8

• Exemple cod SQL

Creare tabelă exemplu

```
CREATE TABLE app_users(
    user_id NUMBER(4) NOT NULL ,
    first_name VARCHAR2(20) NOT NULL,
    last_name VARCHAR2(20) NOT NULL,
    location_id NUMBER(4) NOT NULL,
    email VARCHAR2(30) NOT NULL UNIQUE,
    phone VARCHAR2(20) NOT NULL UNIQUE,
    CONSTRAINT user_id_pk PRIMARY KEY(user_id),
    CONSTRAINT user_location_id_fk FOREIGN KEY(location_id) REFERENCES locations);

CREATE TABLE accounts(
    user_id NUMBER(4) NOT NULL,
    username VARCHAR2(30) NOT NULL UNIQUE,
    password VARCHAR2(100) NOT NULL,
    account_type VARCHAR2(10) NOT NULL,
    CONSTRAINT account_id_pk PRIMARY KEY(user_id),
    CONSTRAINT account_user_id_fk FOREIGN KEY(user_id) REFERENCES app_users,
    CONSTRAINT account_user_pass_ch CHECK (length(password) >= 6)
);
```

Inserare în tabelă exemplu

```
INSERT INTO orders (user_id, shipping_id, product_id, quantity, total_amount) VALUES
((SELECT user_id from app_users where email='andrei.dorcu@yahoo.com'),
(SELECT shipping_id from shipping_methods where provider='Fedex' and delivering_price=65),
(SELECT product_id from products where product_name='Covor Traditional' and shop_id =
(select shop_id from shops s, locations l where s.shop_name = 'SA Covoare Ungheni' AND l.location_id = s.location_id AND l.street_address = 'Str. Mihai Eminescu' AND l.city = 'Ungheni' AND l.country='Republica Moldova')),
1,
1 * (select price from products where product_name='Covor Traditional' and shop_id =
(select shop_id from shops s, locations l where shop_name = 'SA Covoare Ungheni' AND l.location_id = s.location_id AND l.street_address = 'Str. Mihai Eminescu' AND l.city = 'Ungheni' AND l.country='Republica Moldova'))
+ (SELECT delivering_price from shipping_methods where provider='Fedex' and delivering_price=65));

INSERT INTO accounts (user_id, username, password, account_type) VALUES ((SELECT user_id from app_users where email='marin.moisii@yahoo.com'), 'marin', 'moisii', 'admin');

INSERT INTO shipping_methods (provider, delivering_price) VALUES ('China Post Registered Air Mail', 1.8);

INSERT INTO app_users (first_name, last_name, location_id, email, phone) VALUES ('Alex','Ilioi',
(SELECT location_id FROM locations WHERE street_address='Str. Tudor Vladimirescu' AND city='Iasi' AND country='Romania'),
'alex.iliioi@yahoo.com', '0701234568');
```

```
def login(self):
    username = self.username_entry.get().replace('\n', '\n')
    password = self.password_entry.get().replace('\n', '\n')

    #-----Use encryption when sending data across internet
    pass_encrypted = Cipher.encrypt(password)
    log.info("Password encrypted: {}".format(pass_encrypted.decode()))
    password = Cipher.decrypt(pass_encrypted)
    log.info("Password decrypted: {}".format(password))
    #end encryption and decryption part

    query = "SELECT u.user_id, u.first_name, u.last_name, u.location_id, u.email, u.phone, a.account_type from app_users u, accounts a where u.user_id = a.user_id and a.username('{}') and a.password('{}')".format(
        username, password)
    user_info = [item for t in self.controller.run_query(query) for item in t]
    if user_info:
        self.controller.user_info['user_id'] = user_info[0]
        self.controller.user_info['first_name'] = user_info[1]
        self.controller.user_info['last_name'] = user_info[2]
        self.controller.user_info['location_id'] = user_info[3]
        self.controller.user_info['email'] = user_info[4]
        self.controller.user_info['phone'] = user_info[5]
        self.controller.user_info['user_level'] = user_info[6]

        self.controller.re_create_frames()
        self.set_states(user_info[6])

        self.controller.frames["HomePage"].home_page_welcome_label_var.set("Welcome {}".format(user_info[1]))
        self.controller.frames["HomePage"].populate_the_table_with_all_values()
        self.controller.show_frame("HomePage")
    else:
        log.info("Login Failed Incorrect username as password")
        from tkinter import messagebox
        messagebox.showinfo("Login Failed", "Wrong username or password")
```

```
def insert(self):
    log.info("Insert product Page")
    price = self.price_insert.get()
    shop_id = self.shop_id_insert.get()

    name = self.product_name_insert.get()
    if not self.string_length_is_okay(name, text="Product Name"):
        return
    name = name.strip()

    description = self.description_insert.get()
    if not self.string_length_is_okay(description, text="Description Name", length=100):
        return
    description = description.strip()

    if not self.is_number(price):
        from tkinter import messagebox
        messagebox.showinfo("Insert Error", "Price is not number")
        return
    if not shop_id.isdigit():
        from tkinter import messagebox
        messagebox.showinfo("Insert Error", "Shop Id is not number")
        return
    if not self.exist_shop_id(shop_id):
        from tkinter import messagebox
        messagebox.showinfo("Insert Error", "Shop Id does not exist")
        return

    if self.fields_are_empty(name, price, shop_id) or self.product_exists(name, price, shop_id, description):
        return

    name = name.replace('\n', '\n')
    description = description.replace('\n', '\n')

    insert_query = "INSERT INTO products (product_name, price, shop_id, description) VALUES ('{}', '{}', '{}', '{}')".format(name, price, shop_id, description)
    self.controller.run_query(insert_query)
```